AD-A104 987    ARIZONA UNIV  TUCSON  DEPT OF AEROSPACE AND MECHANICA--ETC  F/G 9/2
SOLUTION OF NONLINEAR STRUCTURAL PROBLEMS USING ARRAY PROCESSOR--ETC(U)
AUG 81  N SARIGUL, J MAITAN, H KAMEL                    N00014-75-C-0837

UNCLASSIFIED  TR-8                                                    NL

1 OF 1
AD
A104987

END
DATE
FILMED
10-81
DTIC

N00014-75-C-0837

# SOLUTION OF Nonlinear STRUCTURAL PROBLEMS USING ARRAY PROCESSORS

M. SARIGUL, J. MAITAN, H. KAMEL
University of Arizona
Aerospace and Mechanical
Engineering Department
Tucson, Arizona 85721

August 20, 1981

Technical Report No. 8

Department of the Navy
Office of Naval Research
Structural Mechanics Program (Code 474)
Arlington, Virginia 22217

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

| REPORT DOCUMENTATION PAGE | | READ INSTRUCTIONS BEFORE COMPLETING FORM |
|---|---|---|
| 1. REPORT NUMBER: Technical Report No. 8 | 2. GOVT ACCESSION NO.: AD-A104 | 3. RECIPIENT'S CATALOG NUMBER: 987 |
| 4. TITLE (and Subtitle): SOLUTION OF STRUCTURAL PROBLEMS USING ARRAY PROCESSORS, | | 5. TYPE OF REPORT & PERIOD COVERED: Technical 8/18/81 |
| | | 6. PERFORMING ORG. REPORT NUMBER |
| 7. AUTHOR(s): Sarigul, Nesrin; Maitan, Jacek; Kamel, Hussein | | 8. CONTRACT OR GRANT NUMBER(s): N00014-75-C-0837 |
| 9. PERFORMING ORGANIZATION NAME AND ADDRESS: University of Arizona, Aerospace & Mechanical Engineering Dept., Tucson, Arizona 85721 | | 10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NR 064-531/12-17-75 |
| 11. CONTROLLING OFFICE NAME AND ADDRESS: Dept. of the Navy, Office of Naval Res. Structural Mechanics Program (Code 474) Arlington, Virginia 22217 | | 12. REPORT DATE: August 20, 1981 |
| | | 13. NUMBER OF PAGES |
| 14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office): Same as above | | 15. SECURITY CLASS. (of this report): Unclassified |
| | | 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE |

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Presented at FENOMECH '81, Stuttgard, West Germany
August 25-28, 1981

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Finite element analysis, Computer algorithms, Multiprocessing, Array processors.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The paper describes current efforts to investigate the usefulness of so called "array processors", combined with mini- and superminicomputers, in the analysis of computationally demanding engineering problems, such as in finite element applications.

An array processor, also called an attached processor, is a high speed special purpose computational device, which can per-

[illegible]

form repetitive computations on well structured data sets at effective speeds far beyond those achieved by minis and superminis Although their speeds are substantially below the newest generations of large scale pipeline and parallel vector computers, they do provide a clear cost/performance advantage. Furthermore, the combination of a minicomputer and an array processor provides a high degree of interactivity in addition to attractive computational speeds.

Using an array processor/minicomputer combination effectively is a difficult task, requiring not only knowledge of pertinent algorithms and computer programming, but also a clear understanding of the hardware and the details of its operation. The objective is to produce efficient implementation of practical algorithms. These implementations, however, should eventually be user transparent, in order to be of benefit to the engineering community.

The array processor/minicomputer combination is a special case of a multiprocessor system in which only two processors, with widely differing characteristics, are involved. One interesting factor here is that the communication between the two processors leaves a substantial amount of control in the hands of the programmer. Experimentation with such a system provides an insight into the more general area of parallel processing. The issues here relate to the synchronization of computations and data transfers, as well as, the design of operating systems and applications programs to handle the solution of complex problems.

Several operations typical of finite element analysis were chosen as a basis for performance measurements. Some experiments measured the speeds of execution on the host computer and the array processor, acting as separate processors. Other measurements were designed to assess the performance of the integrated system. In addition to these measurements, simulation was used to predict the performance. The simulation results are matched to the measured results in order to gain confidence in the simulation procedures. Once this has been accomplished, the same methods are used to predict the performance more generally.

Measurements were taken for simple matrix operations, numerical integration of solid element properties, and the decomposition of a hypermatrix. In the case of the computation of stiffness matrices for the solid element, several alternative strategies were attempted, some of which were designed to minimize data transfers between the host and the array processor.

Finally, the experience gained so far is used to examine the possibility of implementing several algorithms typical of the effectiveness of the system in execution of such computations is made.

# SOLUTION OF NONLINEAR STRUCTURAL PROBLEMS USING ARRAY PROCESSORS

| | |
|---|---|
| SARIGUL, NESRIN | Grad. Res. Assistant |
| MAITAN, JACEK | Systems Analyst |
| KAMEL, HUSSEIN A. | Professor |

University of Arizona, Aerospace and Mechanical Engineering Dept.

## SUMMARY

The paper describes ongoing efforts to investigate the usefulness of so called "attached processors" or "array processors", combined with mini- and superminicomputers, in the analysis of computationally demanding engineering problems; such as, in finite element nonlinear applications. A close examination of some basic algorithms, which play an important part in such analysis, is presented. Furthermore, a methodology is developed, which forms a useful guide to further efforts of this nature, including the more general case of multiprocessing. An emphasis is made on the ability to predict the performance of algorithms from time measurements of certain basic operations, coupled with an understanding of the characteristics of the hardware in question and the interplay between its various components. Four hardware alternatives are considered, two of which have attached array processors. Of the algorithms considered, it appears that the decomposition process is the one which benefits most from the presence of an array processor. The speed advantage gained in stiffness assembly, and forward and backward substitution may both be obtained by purchasing a more powerful superminicomputer.

## INTRODUCTION

The advent of "super computers", such as the STAR and CRAY series, and the HEP computer [1], provides an undoubted breakthrough for large scale finite element analysis [2]. The availability of inexpensive microprocessors has triggered efforts to devise microcomputer networks, using parallel and distributed processing to provide faster solution of finite element problems [3,4]. Finite element software is moving into a new era, in which the proper programming environment and modular program systems [5] are replacing monstrously large programs.

In contrast to efforts based on high performance sixth generation computers [2,6], this paper describes ongoing efforts [7] to investigate the usefulness of so called "attached processors" or "array processors", combined with mini- and superminicomputers, in the analysis of computationally demanding engineering problems; such as, in finite element nonlinear applications. The work presented here does not include actual solutions of such problems. Rather, a close examination of some basic algorithms, which play an important part in nonlinear analysis, is presented. Furthermore, a methodology is developed, which forms a useful guide to further efforts of this nature, including the more general case of distributed processing. An emphasis is made on the ability to predict the performance of algorithms from time measurements of certain basic operations, coupled with an understanding of the characteristics of the hardware in question and the interplay between its various components.

The array processor/minicomputer combination is a special case of a multiprocessor system in which only two processors, with widely differing characteristics, are present. Experimentation with such a system provides an insight into the more general area of parallel processing. The issues here relate to the synchronization and balancing of computations and data transfers, as well as the design of operating systems and applications programs to handle the solution of complex problems [8,9,10].

Measurements taken on an array processor/minicomputer system show that, while the array processor is perhaps 80 times faster than the host 16 bit minicomputer in the performance of basic matrix arithmetic, it is often not possible to apply this computational power to practical problems, due to the various restrictions imposed by the limited user address space. An alternate system is now being put together in which a 32 bit minicomputer, with a large address space and a faster data transfer rate, is used as the host.

Several algorithms typical of finite element analysis were chosen as a basis for performance measurements. Some experiments measured the speeds of execution on the host computer and the array processor acting as separate processors. Other measurements were designed to assess the performance of the integrated system.

Measurements were taken for simple matrix operations, numerical integration and assembly of solid element properties, the decomposition, and forward and backward substitutions of a partitioned system of linear equations. In the case of the computation of stiffness matrices for the solid element, several alternative strategies were attempted, some of which were designed to minimize data transfers between the host and the array processor. The experience gained from the current system is used to examine the possibility of implementing these algorithms, which we believe are typical of nonlinear finite element analysis, on the more powerful 32-bit minicomputer system. An assessment of the effectiveness of the alternate system in execution of such computations is made. Some caution has to be exercised in the assessment of the performance of the 16-bit minicomputer based system. Although some of the predicted times for large problems may appear competitive, one should not forget that severe restrictions are placed on problem size due to address limitations, rendering these figures purely hypothetical, except in some dedicated systems.

NOMENCLATURE

It is useful to introduce some symbols and abbreviations, at this point, in order to describe alternate computer configurations, and refer to problem parameters. These abreviations are:

| | |
|---|---|
| HC16 | Refers to 16-bit minicomputer used independently or in conjunction with an array processor. It is characterized by a limited user address space (typically 64 KB) and a slow data bus (typical speed up to 1.5 MB/sec). |
| HC32 | Alternate 32-bit superminicomputer used independently or with an array processor. Since the array processor is assumed to provide the primary computational tool, the selected HC32 system has the advantages of a 32-bit mincomputer of a large address space (16 MB) and a fast data bus (speed 26.5 MB/sec). On the other hand, its floating point processing speed is somewhat limited (0.435 MFLOPS in Whetstone tests, as opposed to the HC16 time of 0.190). Todays popular superminis have been measured at 1.2 MFLOPS, and the top of the line of our HC32 series has been measured at 3.5 MFLOPS. |
| AP | Stands for array processor. The one used in this research is primarily a double precision device, capable of performing 64 bit vector arithmetic, and other well structured computations, at speeds well above those of mini- and superminicomputers. So far, however, it has been used in the |

single precision (32-bit) mode, due to address space limitations. A single precision variant of the same array processor perform the same functions faster and are considerably less expensive.

| | |
|---|---|
| HC16+AP | Combination of the HC16 and AP. The devices communicate via a slow interface, using the HC16 low speed data bus extensively. |
| HC32+AP | Combination of HC32 and AP, which are coupled via a high speed direct memory interface. |
| U | Total number of unknowns in the problem being solved. |
| b | Half bandwidth of problem. Based on element connectivity. |
| S | Submatrix size used to partition the system of equations. |
| P | Total number of partitions in the stiffness matrix. |
| H | Number of non-zero submatrices, per row, in stiffness matrix. |
| M | Number of longitudinal stations in the solid model being analyzed. |
| N | Number of nodes across the solid model in either direction. |
| E | Total number of 8 noded isoparametric solid elements used to model the solids problem. |

## HARDWARE CHARATERISTICS

An array processor, also called an attached processor, is a high speed special purpose computational device, which can perform repetitive computations on well structured data sets at effective speeds far beyond those achieved by minis and superminis. Although their speeds are substantially below the newest generation of large scale pipeline and parallel vector computers, they do provide a clear cost/performance advantage. Furthermore, the combination of a minicomputer and an array processor offers a high degree of interactivity in addition to attractive computational speeds.

The array processor is a separate computer, running under an independent operating system. Its software is primarily suitable for certain types of repetitive operations, such as matrix operations and the computation of Fast Fourier Transforms. It is fairly difficult to incorporate detailed logical branching or computational sequences involving extensive decision making. Data transfers and message swapping between the host and the array processor are problematic and can affect performance appreciably, as seen from some of the measurements taken. The amount of AP local memory may be limited in certain instances. This, however, does not apply to the system under consideration.

Several AP types are available today. We restrict the discussion to the specific device used in this research which, we believe, represents the state of the art. In order to understand the operation of the system, Fig. 1 explains the two basic components, and the various software items which control it, as well as the location of the data. The operating system of the host computer controls the operation of the host, including the user program, which specifies the sequence of events necessary to perform the computation. A special piece of software, called the DRIVER, resides in the host computer. It facilitates communication with the array processor by translating user FORTRAN calls into small packages called Function Control Blocks or FCBs, which are shipped to the array processor to initiate specific computations or data transfers on the latter device. The format of a typical FCB, which occupies six 16-bit words (HC16) or three 32-bit words (HC32), is given in Fig. 2. It contains a function code and operand addresses, which take the form of array processor buffer numbers, as well as some control variables (checksum, etc.).

On the array processor side, an independent operating system, called the EXEC, resides. Its function is to control the operation of the array processor itself. In the array processor there exists a mathematical library, consisting of a number of unlinked (relocatable) routines, which may be assembled into any program being executed inside the array processor. One of the functions of EXEC is to link such programs, as needed, to the specifications contained in the FCBs, and control their execution within the array processor. Such programs take the form of two separate, but fully synchronized, programs called the APU and APS programs respectively. Data are stored within the array processor while being operated upon in special areas called buffers. One important consideration for the system at hand is that, in the FCB, a restricted number of bits is used to store the buffer number. This poses a restriction on the number of buffers which may be defined. At the moment, the maximum number of buffers is 64. Soon it will be expanded to 512. The hardware configurations for the HC16+AP and the HC32+AP are given in Fig. 3 and Fig. 4 respectively.

## BASIC MEASUREMENTS

In attempting to assess the effectiveness of a particular hardware system, such as the one being considered here, several approaches are possible. One approach is to run specific problems on an existing system. This is possible if the system is available, but has the disadvantage of providing specific figures for specific computations and algorithms, *making it difficult to interpret and generalize* [11]. The alternative taken here is to identify specific algorithms pertinent to the subject of the paper, and to use these to identify the basic computational building blocks, which the system has to perform. Careful measurements are conducted, using the hardware whenever possible, to obtain realistic figures for the time taken by each process for various cases. These measurements are then used to obtain closed form expressions with which one can interpolate, or extrapolate, to cover other parameter values. A simple simulation of the algorithm is then used to obtain operation counts for each basic task from which performance estimates may be made. It is possible then to run large numbers of parametric studies, in order to assess the effect of different problem parameters, as well as the effectiveness of the hardware for certain algorithms. These overall estimates are then verified by actual computations of realisitic problems.

Four systems have been considered, HC16, HC32, HC16+AP and HC32+AP. So far the performance of the HC16 and HC16+AP have been measured. The performance figures for the other two systems have been estimated, since they have not been fully installed yet. It is important to note that both host CPU and elapsed times, associated with each operation, have been measured. However, only elapsed times are presented. By doing this, it is implied that performance is measured by the clock time needed to execute the algorithm rather than by the CPU time, which can be deceptive. It is also assumed that a system, such as the one examined here, is intended primarily as a stand alone high performance system, even though it may well be capable of time sharing and multiprocessing.

Another issue to consider is that all measurements and storage space considerations are for single precision (32-bit) computations, even though the array processor is a double precision system, which has also single precision capabilities. We expect to extend the measurements later to double precision. Many of the conclusions reached here, however, may be generalized or extended, using reasonable assumptions. Charging to double precision will impact primarily the HC16, since the address space is quite limited. Execution speeds will also affect the host processor more than the AP, since the AP is, basically, a double precision device. It is also important to note that if single precision were adequate, less expensive array processors are available, which would be much faster.

The first type of operation of significance is that of FCB transfers. The time for that is fixed, and depends primarily on the interface. For the HC16+AP interface, it has been measured at approximately 8.5

milliseconds. Due to the superior architecture of the HC32+AP interface and the simpler associated software, it is estimated that the time will be reduced to less than 1 millisecond.

It is assumed that problem data will be stored on a disk. The speed of such transfers depends on many factors. If one assumes that the system is dedicated to the computation, the elapsed time depends on disk latent time, transfer speed, physical data structure, and data bus speed. Formulas for both systems were obtained. The HC16 formula is based on measurements. Estimates for the HC32 are based on careful assessment of hardware and operating system differences.

Another type of data transfer is that between the host computer and the array processor memories. This transfer is only necessary in the HC16+AP system, since the common memory interface eliminates this operation in HC32+AP. For rather large bandwidths, where the array processor memory may not be sufficient to hold the required data, the host memory may be used as a buffer, and this value will again play a part in the estimate. However, in the case of HC32+AP, this is practically a direct memory transfer, which takes place at memory speed with a small overhead. Figures 5.a and 5.b show plots of measured and estimated data transfer speeds for HC16+AP.

Matrix multiplication represents the major computational effort in most algorithms considered here. Matrix inversion and addition also play some part in the computation. Formulae for estimating elapsed times for such operations, in both host computers and in the AP, have been obtained. Figures 6 and 7 show time estimates with a comparison of the relative speeds of host computer and AP. It has been observed that speed ratios of up to 80 are possible for large matrices. Examples of basic task times for a 40X40 matrix are given in Table 1.

STIFFNESS COMPUTATION AND ASSEMBLY

The process of stiffness assembly is a basic and important one for both linear and nonlinear analysis. Here, two basic functions are performed, element stiffness computation and assembly into the master stiffness matrix. For the purpose of this investigation, the solid model shown in Fig. 8 was chosen. By varying the number of stations (M) and the number of nodes in each direction of the cross-section (N), different combinations of problem size and bandwidth are produced, which form the basis for a parametric study. One important assumption made was that the core memory must be large enough to contain the shaded area of the stiffness matrix, shown in Fig. 8. In this way, the submatrices of the stiffness hypermatrix are generated in core and written to the disk only once.

At this point, it is appropriate to bring up the issue of problem size limitations, which apply here equally for stiffness assembly and decomposition. Fig. 9 illustrates this point. Several factors limit problem size , and these constraints are represented by straight lines or curves in the figure. For example, the effect of the limited number of buffers is shown by vertical lines. At present, the system has a limitation of 64 buffers, restricting the maximum number of submatrices per row to 9. It is soon expected that a new software release will relax this to 512, allowing 31 submatrices per row. The size of memory 1 in the AP is currently equal to 16 KW, minus 11 KW needed for EXEC and the mathematical library. Since in the current decomposition routine 5 submatrices are placed in this memory, a submatrix size of 30X30 can not be exceeded. If the memory is expanded to 32 KW, it is possible to employ 62X62 matrices; and if extended to 64 KW, matrices over 100X100 may be handled. With the total space available on AP memories 2 and 3, a maximum half bandwidth b, of approximately 320, can be handled. This is illustrated by the first of a set of curves showing the possible combinations of S and H which may be handled. If the two memories are expanded to their full capacity, b may be increased to 450. If larger problems are to be tackled, the host computer memory has to be used as additional buffer space. If the memory is expanded to 1 MB, b may be increased to 1400. If the memory is increased to 2 MB, the maximum b is equal to 2000. The use of host memory as a buffer may be done efficiently in HC32 for reasons sited above. As for HC16, the address space restrictions would prevent that. Even if the HC16 computer did not have such a restriction, memory transfer times with a conventional interface may become excessive [7]. From the above, it appears that, for the current HC16+AP configuration, a maximum of b=320 may be achieved. If the configuration is fully expanded b may increase to 450. To solve larger problems effectively, the HC32+AP is a necessity.

In evaluating the elapsed time necessary for performing the stiffness computation, two items must be considered, one relating to the computation and numerical integration of the individual elements, and the other is that of transferring the assembled submatrices onto the disk. Three different schemes for the former operation were considered. In the selected scheme, all data preparation, including the Jacobian and its inverse along with the preparation of the shape function derivatives, are performed in the host. It also performs the element assembly. The strain matrices are shipped to the AP in condensed form, and the numerical integration is performed there in one single multiplication using a specially written subroutine. This proved to be the most promising arrangement and it was chosen as the working algorithm. The estimated elapsed time for this scheme is 0.522 seconds/element for the HC16+AP and 0.469 for the HC32+AP systems.

In addition to the time needed for in-core computation and assembly, a certain number of submatrix transfers to and from disk are required during the assembly process. The time required for these transfers was also computed and included in the estimate. Results from various runs show that the stiffness computation process is speeded up by a constant factor of approximately 5 for the HC16, and 3.5 for HC32, regardless of bandwidth, by adding the array processor. This result is somewhat disappointing perhaps, but similar conclusions have been reached by others [2]. Table 2 gives the operations count and timing for a realistic (20X12X12) problem where the number of unknowns is 8650 and the half bandwidth is 900.

MATRIX DECOMPOSITION

The problem of matrix decomposition was handled in a similar manner to that of the stiffness assembly. The operations count is given in Table 3 for the same problem discussed previously. In this instance, however, there is an appreciable speed-up due to the use of the array processor. In the case of HC16+AP, the speed up factor is 65 and for the HC32+AP it is 37.5. The time needed to decompose such a matrix, however, is still over two hours.

In order to verify performance predictions, several actual runs were made. Due to the restrictions imposed by the current HC16+AP system, a maximum of 9 submatrices, per row, and a submatrix size of (30X30) were used. In spite of that, problems of a size up to 1200 degrees of freedom and a half bandwidth of 270 were executed. Measurements of the elapsed time were taken, and compared with the results obtained by the simulation program. The results are given in Table 4, and Fig. 10. Table 4 reflects several interesting points. Due to discrepencies between the decomposition program and the optimum algorithm, some differences in the counts were observed. In order to remove the effect of the discrepency, a correction was applied, and both the corrected and uncorrected figures are given. In both cases, the errors are well within the accepted limits, particularly when the measured times can, themselves, vary to the same order of magnitude.This is illustrated by one of the cases, which was run twice and shows a difference of approximately 10 percent. Estimates, on the whole, seem to be more accurate for narrow banded matrices, which is not surprising since this was one of the basic assumptions made in the operation count estimates. The adjusted run times are more accurate than the unadjusted ones. Fig. 10 gives a pictorial representation of the results for a set of decompositions where the bandwidth is kept at a constant value of 270. The results give a certain amount of confidence in the basic approach used in this paper and allows us to extend the estimates to other algorithms.

## FORWARD AND BACKWARD SUBSTITUTIONS

The same procedure applied so far to the stiffness assembly and decomposition is next employed to obtain estimates of elapsed times for the forward and backward substitutions typical of nonlinear iterations. The results, for the same solid model described above, are given in Table 5. They show a definite, yet small, advantage for the HC32+AP system over the HC32. One problem seems to be the ineffectiveness of simple additions inside the array processor. In that case, it might be advantageous to perform this part of the computation inside the host computer. This would speed-up the process somewhat, giving a total speed up factor of 5.23 over the HC32 alone. At this point in time, it appears that the advantage of the array processor is not as substantial here. However, careful measurements are required before final conclusions are made.

## ON THE ADVANTAGES OF PARALLEL COMPUTATION

In a system incorporating multiple asynchronous devices, the question comes to mind as to the possible advantages of parallel computation. Whether an algorithm can be accellerated by allowing the different hardware modules to operate independently, performing different but related functions at the same time, depends on both the algorithm and the hardware configuration [7]. Fig. 11 shows a hypothetical case for which parallel processing is to be applied to a specific algorithm. Three devices are considered, a host computer, a data transfer bus, and an array processor. Fig. 11.a illustrates the sequence of operations, in real time, for the given system, assuming serial computation. The same process may be represented more clearly in Fig. 11.b, where it is evident that the array processor is not being used sufficiently. The host and data bus, on the other hand, are used approximately half the time. The maximum benefit to be derived from parallelism, in this case, is shown in Fig. 11.c, which assumes that all devices can perform the required functions simultaneously. This is usually not possible, however, since operations conducted in one part of the hardware usually requires that other operations conducted elsewhere be completed first, which results in an actual run such as that shown in Fig. 11.d.

Whether a particular algorithm can benefit from parallel computation or not, must first be determined by a simple analysis of the demand on computer resources. If the largest amount o time is spent in one particular device, parallel operation will most certainly not help. It might be possible to shift some of the load from one device to another, in order to obtain a more balanced load on the system. If, for example, an algorithm is compute bound, such as in the case of the decomposition, it is possible to increase the speed by supplying more than one array processor, or CPU, in order to achieve a balance between computation and data transfer times. On the other hand, if the process suffers from excessive data transfer times, data buffering, faster data busses, along with multiple disk drives and controllers may be used. Careful planning of system I/O operation would also help. Once it is determined that an algorithm is potentially suitable for parallel computation, detailed analysis in which the discrete nature of the demands on each system component is taken into consideration, is worth undertaking [7].

## CONCLUSIONS

The paper has succeeded in devising a methodology for predicting the performance improvement as a result of the addition of an array processor to a minicomputer. Four systems were considered, two of which have attached array processors. Of the algorithms considered, it appears that the decomposition process is the one which benefits most from the presence of an array processor. The speed advantage gained in stiffness assembly, and forward and backward substitutions may both be obtained by purchasing a more powerful superminicomputer. As such, it appears that a mix of jobs, in which the host computer is freed to handle other tasks such as pre- and postprocessing along with linearized iterations while the array processor is occupied in the decomposition process, might be advantageous. One might think of alternatives to traditional computational strategies. For example, in the modified Newton method, the host computer may be used to improve on a current solution vector based on a previously obtained decomposition, while the array processor is used to compute and decompose a new stiffness matrix based on a more recent solution point. Using the current time estimates, the advantage gained may be small. Only 6 iterations can be performed in the time required by the HC32+AP to compute a new stiffness and decompose it. Sparse matrix computations may be used for the iterative procedure, rather than a direct solution. It is too early to find out whether such algorithms could be effective on an array processor.

One must state, however, that the apparent advantages of the array processor in the decomposition algorithm, which seems to be the most time consuming of all computational steps, is not to be taken lightly. It is clear that such speeds can not be achieved with simple computer devices with the same price range.

Many other conclusions come to mind. A 16-bit minicomputer with limited address space does not appear to be suitable as a host for a fast array processor. The restricted memory size drastically limits problem size. A conventional communication interface is not adequate if the minicomputer/array processor combination is to be utilized effectively. In using the hypermatrix scheme, small submatrices are to be avoided. A minimum size of 30X30 is recommended. For large problems, stiffness computation and assembly is speeded up by a factor of 3.5, forward and backward substitution is speeded up by a factor of 4 to 5, but matrix decomposition may run as much as 40 times faster. Parallel processing does not seem to offer much of an advantage, unless multiple array processors are used or a mix of jobs is carried on simultaneously by the system. The manufacturing of specialized hardware to perform certain functions effectively is a valid idea and will play an increasingly greater role. Future hardware will probably take the form of a network of such systems.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the support of the Office of Naval Research under contract no. N00014-75-C-0837, CSPI Inc., and the Engineering Experiment Station of the College of Engineering. Special thanks are due to Ms. Susan K. Lewis and Ms. Brenda L. Volp for helping in preparing the manuscript, and to Ms. Bettina Bair and Messrs. Ramesh Kolar and David Zielinski for their help with preparing the figures and slides.

## REFRENENCES

1. Principles of Operations, HEP Technical Documentation Series, Denelcor Pub. #900001, Denver, April 1981.

2. Noor, A.K. and Lambiotte, J.J. jr., Finite Element Dynamic Analysis on CDC STAR-100 Computer, Computers and Structures, Vol. 10, pp 7-19, April 1979.

3. Jordan,H.F. and Sawyer,P.L., A Multiprocessor System for Finite Element Structural Analysis. Computers and Structures, Vol. 10, Nos. 1&2, pp 21-32, 1979.

4. Jordan, H.F. The Finite Element Machine, Programmers Reference Manual. University of Colorado, August 1979.

5. Felippa, C.A. Architecture of A Distributed Analysis Network for Computational Mechanics, Computers and Structures, Vol.13, pp 405-413, June 1981.

6. Grosch, C.E.: Poisson Solver on a Large Array Computer. Proc. of the 1978 LASL Workshop on Vector and Parallel Processors. Los Alamos, New Mexico, 1978.

7. Kamel,H.A. and Maitan,J. Performance of Finite Element Algorithms on an Array Processor-Mini Computer Based System. Nonlinear Finite Element Analysis in Structural Mechanics, Wunderlich, Stein and Bathe (Ed.), Springer Verlag, 1981.

8. Lindt,B. and Agerwala,T. , Communication Issues in the Design and Analysis of Parallel Algorithms, IEEE Transactions on Software Engineering, Vol. SE-7, No. 2, March, 1981.

9. Jones, A.K. et al, Programming Issues Raised by a Multiprocessor. Proc. IEEE, Vol. 66, No. 2, 1978.

10. Robinson, J.T., Some Analysis Techniques for Asynchronous

Multiprocessor Algorithms, IEEE Trans. on Software Engineering, Vol. SE-5, No.1, January 1979.

11. Maitan, J. and Kamel, H.A., Performance of Minicomputers in Finite Element Analysis, Pre- and Postprocessing. ONR Technical Report No. 6, Contract N00014-75-C-0837, U. of Arizona, May 1980.

| ITEM | AP | HC16 | HC32 |
|---|---|---|---|
| HC-DK | | 55.33 | 9.40 |
| HC-AP | | 45.87 | 16.10 |
| Multiplication | 75.58 | 5448.40 | 2724.20 |
| Addition | 10.00 | 65.50 | 32.75 |
| Inversion | 112.34 | 8039.34 | 4019.67 |
| MRB transfer | | 8.50 | 1.00 |
| FCB transfer | | 8.50 | 1.00 |

Table 1. TIMING OF BASIC OPERATION FOR (40X40 SUBMATRICES)
(Times in msecs.)

| ITEM | HC16 | HC32 | HC16+AP | SPEED UP FACTOR | HC32+AP | SPEED UP FACTOR |
|---|---|---|---|---|---|---|
| Transfers | 237.7 | 35.7 | 237.7 | 1.000 | 35.7 | 1.000 |
| Elmnt. Comp. | 7687.9 | 3843.0 | 1200.1 | 6.406 | 1078.2 | 3.565 |
| Total | 7925.6 | 3879.7 | 1437.8 | 5.512 | 1114.0 | 3.483 |
| Parallel | 7687.9 | 3843.9 | 1200.1 | 6.406 | 1078.2 | 3.565 |

NOTE: HC16 times are hypothetical due to memory restrictions.

Table 2. RELATIVE PERFORMANCE FOR TYPICAL SOLID PROBLEM
(STIFFNESS ASSEMBLY) (Times in secs.)

| ITEM | HC16 | HC32 | HC16+AP | SPEED UP FACTOR | HC32+AP | SPEED UP FACTOR |
|---|---|---|---|---|---|---|
| Transfers | 475.5 | 71.5 | 838.6 | 0.567 | 71.5 | 1.000 |
| Multiplication | 555952.1 | 277976.0 | 7144.0 | 77.821 | 7144.0 | 38.910 |
| Addition | 2085.4 | 1042.7 | 143.3 | 14.550 | 143.3 | 7.275 |
| Inversion | 3906.4 | 1953.2 | 51.9 | 75.271 | 51.9 | 37.635 |
| FCB transfer | | | 526.4 | | 61.9 | |
| Total | 562419.3 | 281043.3 | 8704.3 | 64.614 | 7472.7 | 37.609 |
| Parallel | 561943.8 | 280971.8 | 7339.2 | 76.567 | 7339.2 | 38.283 |

Table 3. RELATIVE PERFORMANCE FOR TYPICAL SOLID PROBLEM
(DECOMPOSITION) (Times in secs.)

| NO. of Prtns. | Submatrix size S | N.S.M/row H | U | b/U | Estimated (prog.) | Measure (run) | Pcnt err | Adj. | Pcnt err |
|---|---|---|---|---|---|---|---|---|---|
| 40 | 30 | 9 | 1200 | 0.225 | 217 | 257 | 16 | 233 | 7 |
| 30 | 30 | 9 | 900 | 0.3 | 155 | 179 | 13 | 163 | 5 |
| 20 | 30 | 9 | 180 | 0.45 | 94 | 119 | 21 | 111 | 15 |
| 15 | 30 | 9 | 450 | 0.60 | 63 | 82 | 23 | - | - |
| 15 | 30 | 9 | 450 | 0.60 | 63 | 74 | 15 | - | - |
| 15 | 24 | 9 | 360 | 0.60 | 49 | 51 | 4 | - | - |
| 15 | 20 | 9 | 300 | 0.60 | 42 | 45 | 7 | - | - |
| 15 | 16 | 9 | 240 | 0.60 | 38 | 34 | 12 | - | - |
| 15 | 10 | 9 | 150 | 0.60 | 33 | 31 | 6 | - | - |

Table 4. COMPARISON of MEASURED and PREDICTED TIMES (MATRIX DECOMPOSITION) (Times in secs.)

| ITEM | HC16 | HC32 | HC16+AP | SPEED UP FACTOR | HC32+AP | SPEED UP FACTOR |
|---|---|---|---|---|---|---|
| Tarnsfers | 934.3 | 140.5 | 1647.8 | .567 | 140.5 | 1.000 |
| Multns. | 2564.6 | 1282.3 | 64.9 | 39.540 | 64.9 | 19.770 |
| Addition | 30.9 | 15.5 | 83.3 | .371 | 83.3 | .185 |
| Inversion | .0 | .0 | .0 | .000 | .0 | .000 |
| FCB transfer | | | 416.7 | | 49.0 | |
| TOTAL | 3529.8 | 1438.2 | 2212.7 | 1.595 | 337.7 | 4.259 |
| PARALLEL | 2595.5 | 1297.8 | 2064.5 | 1.257 | 189.5 | 6.848 |

Table 5. RELATIVE PERFORMANCE FOR TYPICAL SOLID PROBLEM (FORWARD AND BACKWARD SUBSTITUTIONS) (Times in secs.)

## FIGURE CAPTIONS

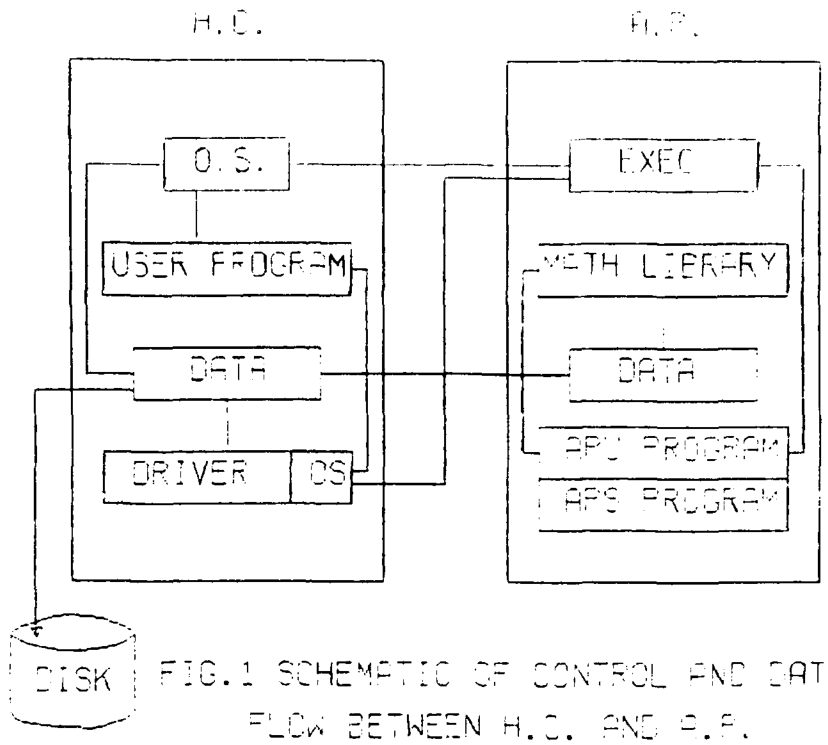

FIG.1 SCHEMATIC OF CONTROL AND DATA FLOW BETWEEN H.C. AND A.P.

| CHECK SUM | |
|---|---|
| FUNCTION CODE | CONTROL 1 |
| *OP 1 | Ø |
| *OP 2 | Ø |
| *OP 3 | Ø |
| *CONTROL 2 | Ø |

FIG. 2 FCB FORMAT

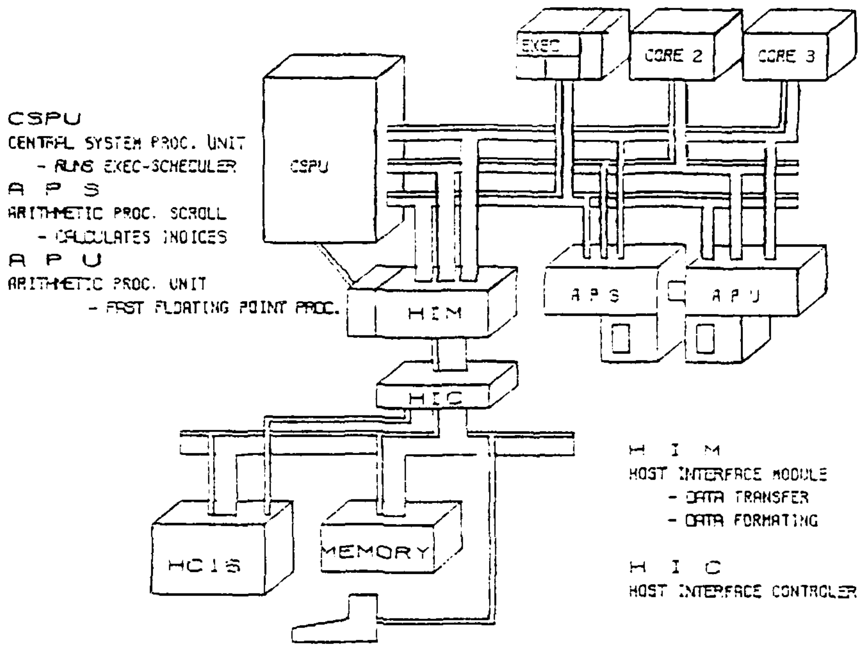

FIG.3 SYSTEM WITH 16 BIT HOST

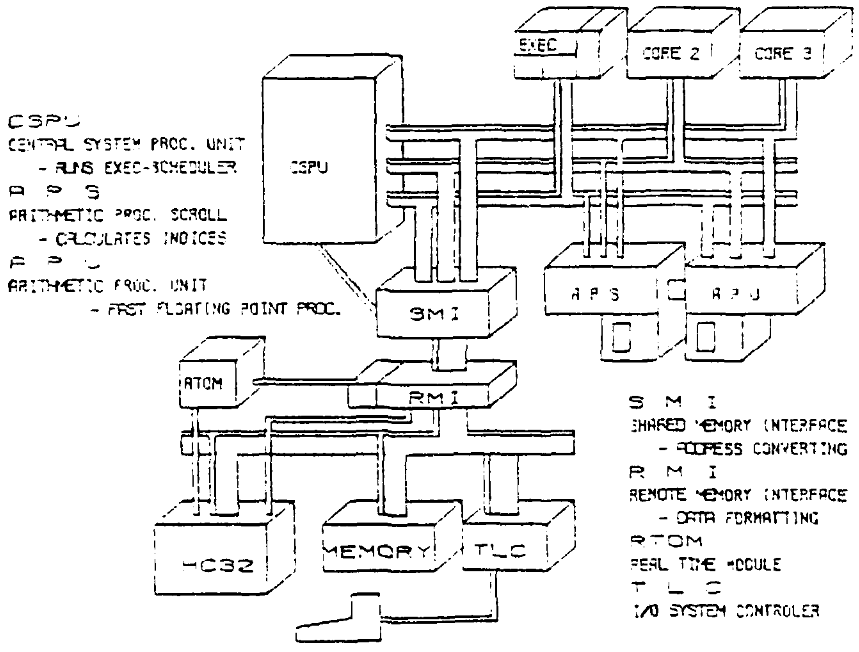

FIG.4 SYSTEM WITH 32 BIT HOST

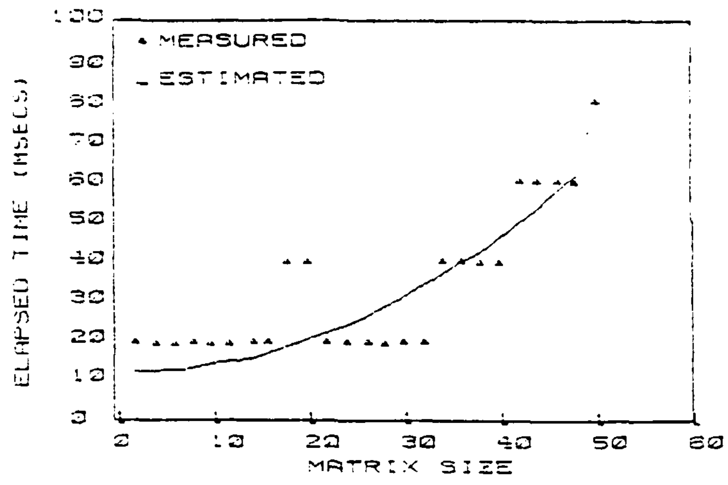

FIG. 5A DATA TRANSFER SPEED
(HC16 - AP)

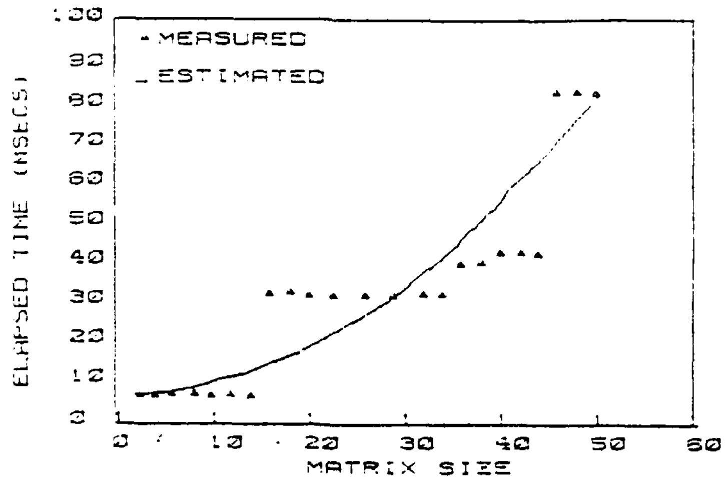

FIG. 5B DATA TRANSFER SPEED
(HC16 - CK)

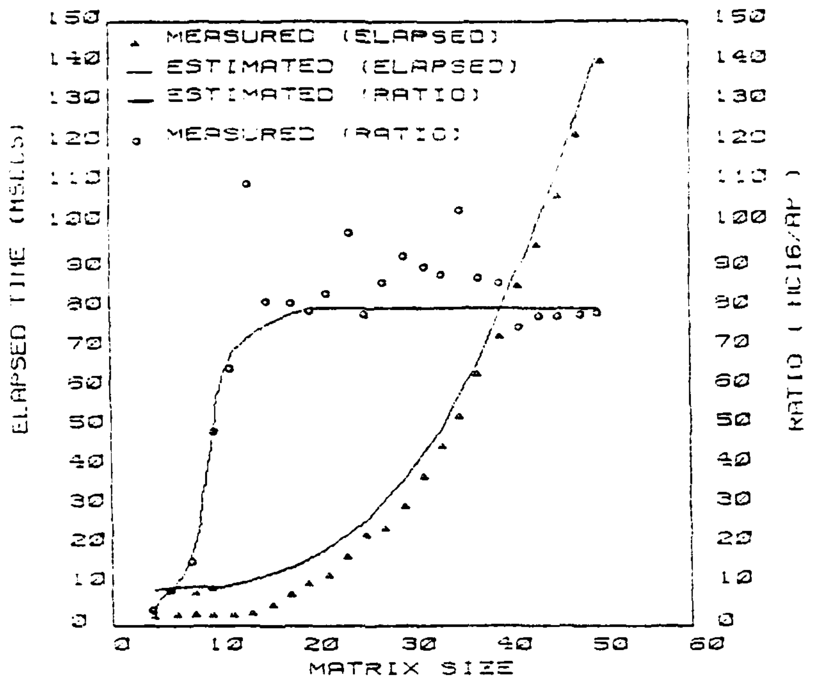

FIG. 6 ARRAY PROCESSOR PERFORMANCE
(MATRIX MULTIPLICATION TIME)

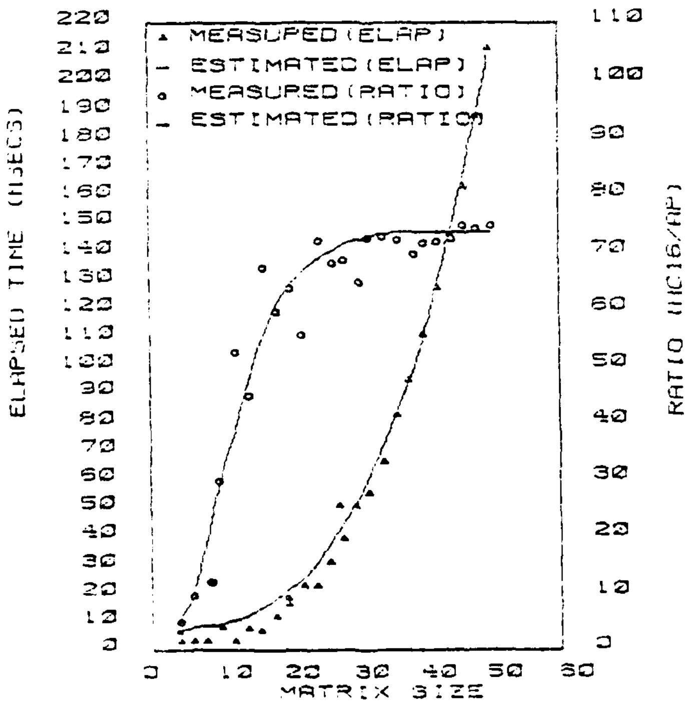

FIG. 7 ARRAY PROCESSOR PERFORMANCE
MATRIX INVERSE

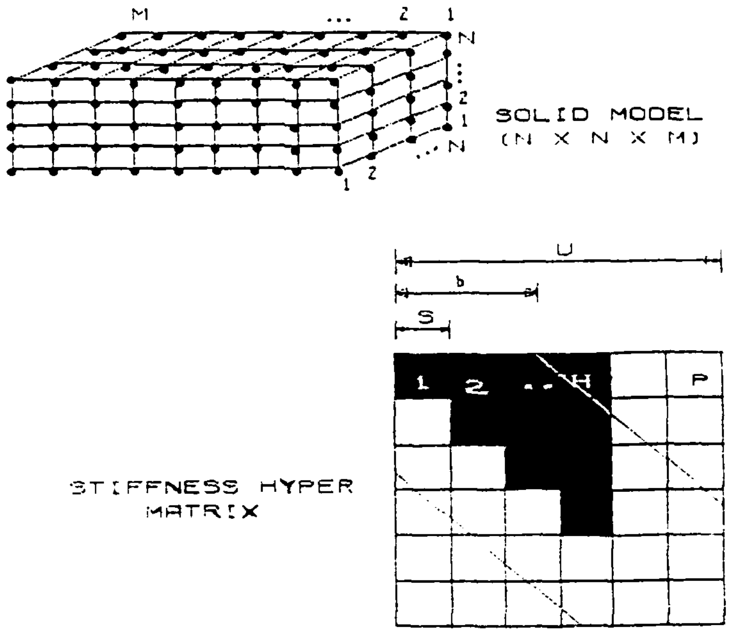

FIG.8 SOLID MODEL EXAMPLE

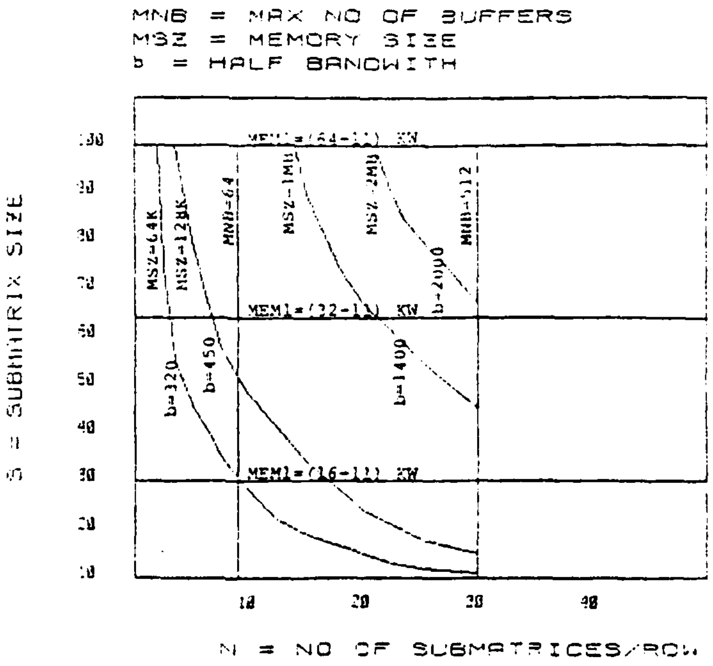

FIG.9 PROBLEM SIZE LIMITATION
( SINGLE PRECISION )

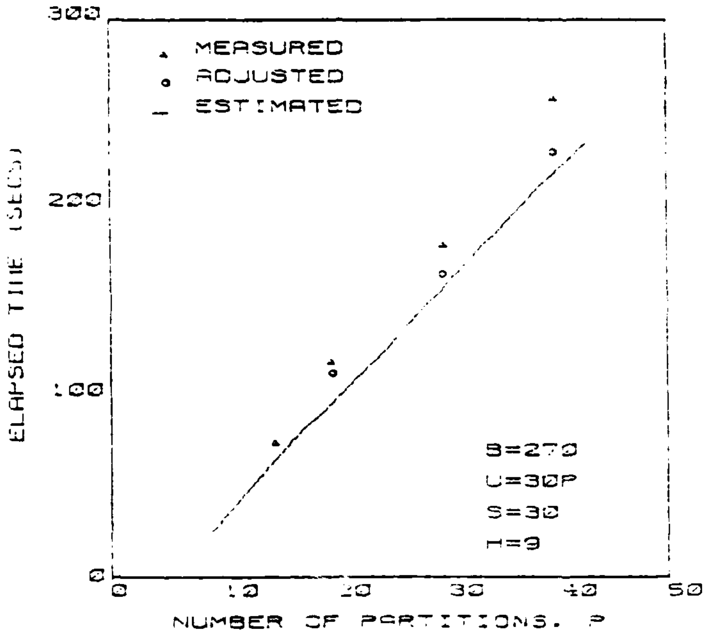

FIG. 10 MATRIX DECOMPOSITION ON HC16+AP
ESTIMATES VS MEASUREMENTS

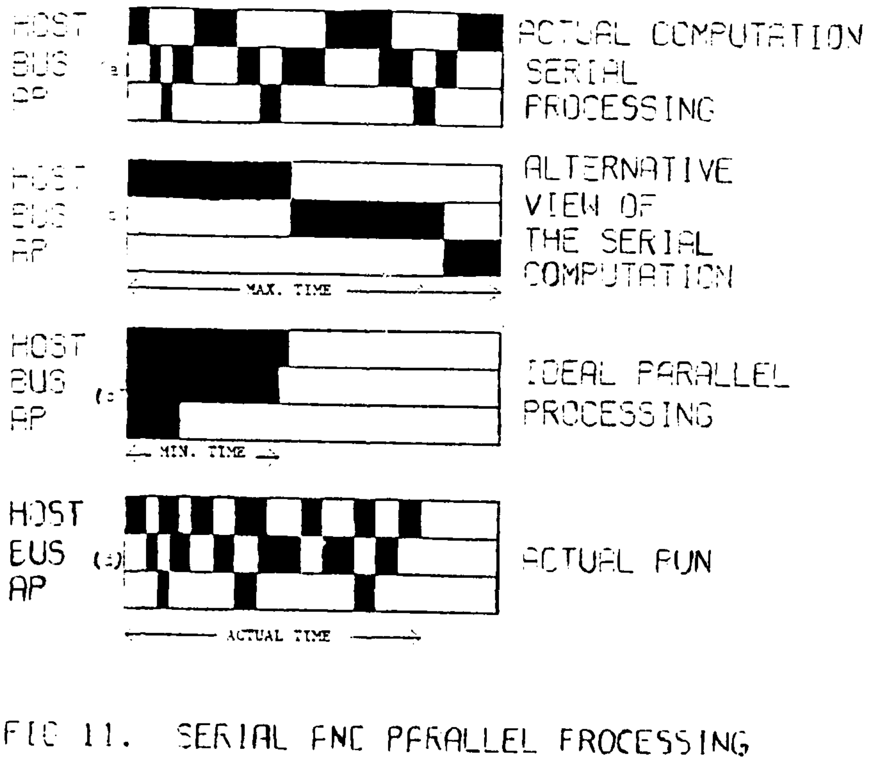

FIG 11. SERIAL AND PARALLEL PROCESSING